Dual Resource Constrained Scheduling Considering Operator Working Modes and Moving in Identical Parallel Machines Using a Permutation-Based Genetic Algorithm

Muhammad Akbar, Takashi Irohara

Dual resource constrained scheduling considering operator working modes and moving in identical parallel machines using a permutation-based genetic algorithm

Muhammad Akbar[1[0000-0001-5736-4313]] and Takashi Irohara[1[0000-0003-2040-9703]]

[1] Department of Information and Communication Sciences, Sophia University, 7-1 Kioi-Cho, Chiyoda-Ku, Tokyo 102-8554, Japan
muhammad@eagle.sophia.ac.jp; irohara@sophia.ac.jp

Abstract. This paper proposes a novel dual resource constrained (DRC) scheduling problem under identical parallel machine environment that consider operator working modes and moving activity between machines with regards to the makespan minimization objective. We define the working modes as all operator activities when the operators interact with the machines such as loading, setup, controlling, and unloading. Firstly, we provide the mathematical model of the problem using Mixed Integer Linear Programming (MILP). We add unloading activity beside setup to be included in the model. Also, we consider the moving activity that is usually neglected in DRC scheduling problem. Moreover, we propose a permutation-based genetic algorithm (PGA) to tackle the computational burden of the bigger size problem. Then, we run a full factorial experiment with replication to compare the solution quality and computational time of our PGA to the solver and random search method. The results show that our proposed PGA could solve the problem in a reasonable time that is faster than the solver with a good quality solution that is better than random search.

Keywords: DRC scheduling, working modes, moving, makespan.

1 Introduction

Recently, many industries use partially automated machines for producing products which operators can leave them in the machining time. The operators are necessary for such only activities like the loading and unloading jobs, sometimes controlling or doing the setup [1]. The total idle time of operators becomes the highest if they only operate a machine each. However, assigning an operator to two or more machines results in some machine waiting time that increases the makespan or the tardiness.

Dual resource constrained (DRC) scheduling considers both machine and operator resources as limiting resources [2]. Most research in DRC scheduling deals in a condition that the operator number is not less than the number of machines, e.g. [3] and [4]. On the next level, we find articles that deal with fewer operators, but they cannot supervise many jobs at the time [5], [6]. Only a little research considers an operator can supervise several machines simultaneously [7] as in [1], [8].

There are several approaches to solve the DRC scheduling with each operator can process many jobs simultaneously. One approach uses a given finite set of working modes as a reference to estimate the job processing time using multipliers [1]. The other approach uses mixed-integer linear programming (MILP) model allowing an operator to do many setup activities and leave the machine while processing the job to minimize the makespan [8]. In this work, we combine the advantages of both to propose a new approach by applying MILP model to get a more accurate solution and considering many working modes such as loading, setup and unloading activities. Also, this research considers transportation time between machines.

2 Problem statement

Figure 1 illustrates the proposed DRC scheduling problem with three identical parallel machines and two same-skill operators with five jobs. It must consider not only the sequence of jobs in each machine but also the activity/task sequence of jobs and moving sequence performed by each operator that makes this problem becomes harder to solve. Therefore, we must include the operators' assignment on the Gantt chart to describe the schedule. The new consideration of task allocation causes the job assignment to the operator becomes more flexible in which one job can be performed by different operators, e.g., setup performed by an operator and unloading by another operator. We only consider setup (that includes loading) and unloading activities in the model. This contribution do not appear in the traditional scheduling problem since the operators always control the same machine. The other contribution related to moving route and time makes the computation of makespan more precise.

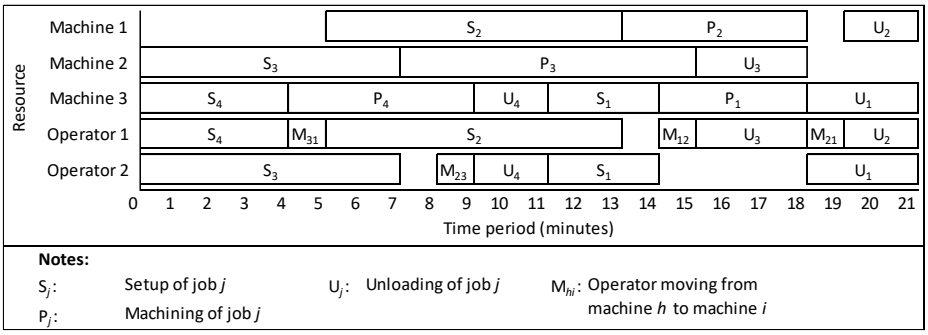

Fig. 1. An illustration of a Gantt chart of the proposed DRC scheduling problem with 5 jobs

We describe the problem statement of this DRC scheduling problem as follows. The production system consists of a set M of m identical parallel machines and a set W of w identical operators where $w \leq m$. This production system must produce a set N of n jobs aiming at the minimization of the makespan. Each job needs three sequence activities, which are setup, machining, and unloading performed on a unique machine. Each operator can contribute to any b activities of working modes from a set B. In our case, the b value is two consisting setup and unloading activities. Thus, each

operator could move from one machine after finishing one activity *b* to the other machine to perform any activity in set *B*. Any activity *b* of job *i* starts if both the assigned machine *j* and operator *k* are available.

3 The mathematical model

We refer to the previous research [8] then adjust and add some new constraints to build our own MILP model. We adjust some constraints in several ways by adding new activity indices (1 for setup and 2 for unloading) to the decision variables and place the operator indices as the main resource instead of the machine. It is because the operator assignment is more complicated than the machine since two operators are possible to execute the same job. We also include transportation time between machines as a new parameter. The mathematical formulation is reported as follow.

Indices

$f, j, l = 0,1,2, \dots, n$	jobs	$k, q = 1,2, \dots, w$	workers
$g, h, i = 1,2, \dots, m$	machines	$a, b, c = 1, 2$	activities

Parameters

P_l	the processing time of job *l*	T_{gi}	the moving time between machine *g* and *i*
O_{bl}	the operation time of activity *b* of job *l*	B	a big number

Decision variables

$X_{khajibl}$	a binary variable that result in 1 if worker *k* is assigned to do activity *b* of job *l* on machine *i* after finishing activity *a* of job *j* on machine *h*.
Q_{fl}	a binary variable that result in 1 if setup activity of job *l* performs before unloading activity of job *f* in the same machine.
O_{bl}^{c}	the operation completion time of activity *b* of job *l*
P_{l}^{c}	the processing completion time of job *l*
T_{bl}^{c}	the transportation/moving completion time before doing activity *b* of job
C_{max}	the makespan

Model

minimize C_{max} (1)

subject to:

$\sum_{k=1}^{w}\sum_{h=1}^{m}\sum_{a=1}^{2}\sum_{j=0}^{n}\sum_{i=1}^{m} X_{khajibl} = 1 \quad \forall b = 1,2;\ l = 1,2, \dots, n$ (2)

$\sum_{k=1}^{w}\sum_{h=1}^{m}\sum_{i=1}^{m}\sum_{b=1}^{2}\sum_{l=1}^{n} X_{khajibl} \leq 1 \quad \forall a = 1,2;\ j = 1,2, \dots, n$ (3)

$\sum_{h=1}^{m}\sum_{i=1}^{m}\sum_{b=1}^{2}\sum_{l=1}^{n} X_{kh20ibl} \leq 1 \quad \forall k = 1,2, \dots, w$ (4)

$\sum_{h=1}^{m}\sum_{i=1}^{m}\sum_{b=1}^{2}\sum_{l=1}^{n} X_{kh10ibl} = 0$ (5)

$\sum_{k=1}^{w}\sum_{h=1}^{m}\sum_{a=1}^{2}\sum_{j=0}^{n} X_{khaji1l} - \sum_{q=1}^{w}\sum_{g=1}^{m}\sum_{c=1}^{2}\sum_{f=0}^{n} X_{qgcfi2l} = 0$
$\forall i = 1,2, \dots, m;\ l = 1,2, \dots, n$ (6)

$\sum_{h=1}^{m}\sum_{a=1}^{2}\sum_{j=0}^{n} X_{khajibl} \geq \sum_{g=1}^{m}\sum_{c=1}^{2}\sum_{f=1}^{n} X_{kiblgcf} \quad \forall k = 1,2, \dots, w;$
$i = 1,2, \dots, m; b = 1,2;\ l = 1,2, \dots, n$ (7)

$O_{bl}^{c} - T_{bl}^{c} \geq O_{bl} \quad \forall b = 1,2;\ l = 1,2, \dots, n$ (8)

$T_{bl}^{c} - O_{aj}^{c} \geq \sum_{h=1}^{m}\sum_{i=1}^{m}\sum_{k=1}^{w} T_{hi} \cdot X_{khajibl} - B \cdot \left(1 - \sum_{h=1}^{m}\sum_{i=1}^{m}\sum_{k=1}^{w} X_{khajibl}\right)$
$\forall b = 1,2;\ l = 1,2, \dots, n; a = 1,2;\ j = 0,1, \dots, n$ (9)

$$O_{2l}^{c} - P_{l}^{c} \geq O_{2l} \quad \forall\, l = 1,2,\dots,n \tag{10}$$

$$P_{l}^{c} - O_{1l}^{c} = P_{l} \quad \forall\, l = 1,2,\dots,n \tag{11}$$

$$\begin{cases} O_{1l}^{c} - O_{2f}^{c} \geq \sum_{k=1}^{w}\sum_{h=1}^{m}\sum_{a=1}^{2}\sum_{j=0}^{n} O_{1l} \cdot X_{khaji1l} - B \cdot \left(2 - \sum_{k=1}^{w}\sum_{h=1}^{m}\sum_{a=1}^{2}\sum_{j=0}^{n}\left(X_{khaji1l} + X_{khaji1f}\right) + Q_{fl}\right) \\ O_{1f}^{c} - O_{2l}^{c} \geq \sum_{k=1}^{w}\sum_{h=1}^{m}\sum_{a=1}^{2}\sum_{j=0}^{n} O_{1f} \cdot X_{khaji1f} - B \cdot \left(2 - \sum_{k=1}^{w}\sum_{h=1}^{m}\sum_{a=1}^{2}\sum_{j=0}^{n}\left(X_{khaji1l} + X_{khaji1f}\right) + 1 - Q_{fl}\right) \end{cases}$$

$$\forall\, i = 1,2,\dots m;\ f = 1,2,\dots,n; l = f+1,\ f+2,\dots,n \tag{12}$$

$$O_{20}^{c} = 0 \tag{13}$$

$$C_{max} \geq O_{2l}^{c} \quad \forall l = 1,2,\dots,n \tag{14}$$

$$X_{khajibl} \in \{0;1\} \quad \forall k = 1,2,\dots,w; h = 1,2,\dots,m; a = 1,2; j = 0,1,\dots,n;$$
$$i = 1,2,\dots,m; b = 1,2; l = 1,2,\dots,n \tag{15}$$

$$Q_{fl} \in \{0;1\} \quad \forall\, f = 1,2,\dots,n; l = f+1,\ f+2,\dots,n \tag{16}$$

Constraint (2) ensures the assignment any activity of specific job to only one operator and one machine, and there is only one activity of one job which precedes. Constraint (3) forces that each activity of one job precedes at most one other activity of one job conducted by the same operator. Constraint (4) denotes each worker starts by unloading of job 0. Constraint (5) ensures that there is no assignment for setup of job 0. Constraint (6) forces that setup and unloading activities of one job must perform on the same machine. Constraint (7) ensures the feasibility of sequence of activity by each operator: if activity *b* of job *l* precedes activity *c* of job *f*, it must have a predecessor of activity *a* of job *j*. Constraint (8) states that the minimum time lag between the operation completion time of and the moving completion time before activity *b* of job *l* must be equal to its operation time. Constraint (9) denotes that the difference between the moving completion time to do activity *b* of job *l* and the completion time of previous activity *a* of job *j* is at least equal to the moving time required by itself. Constraint (10) ensures, on each machine, that the unloading activity is possible after the machine finishes processing the job. Constraint (11) forces that each machine starts processing the job immediately after the setup activity. The twofold constraints (12) accommodates the feasibility sequence on each machine: if setup activity of job *l* and unloading activity of job *f* processed on the same machine, then unloading activity of job *f* must be completed before setup activity of job *l* starts, or vice versa. Constraint (13) fixes there is no time for unloading activity of job 0. Constraint (14) defines the makespan value. Finally, Constraint (15) and (16) define the corresponding binary variables.

4 The proposed permutation-based GA

One of the most common meta-heuristic technique used in scheduling problem is Genetic Algorithm (GA) [9]. Since it is proven and easier to develop, we consider GA to solve our problem at first that can perform as the basic comparator to other techniques for the future research. Genetic algorithm (GA) that is found by Holland in 1975 mimics the biological processes [10]. This meta-heuristic procedure works with a set of solutions (*chromosomes*) called as *population*. A chromosome, which contains alleles, represents one solution that is evaluated by its *fitness* value that indicates the objective function. The initial population usually comes from a random process. At every iteration, the *selection* mechanism chooses a couple of chromosomes

(*parents*). The parents follow the *crossover* mechanism to generate new solutions (*offspring*) by combining their structure. Moreover, each chromosome can experience changing of allele sequence based on the *mutation* scheme to prevent the solution trapped into local optima. Finally, this algorithm will stop to iterate after it meets the stop criterion such as the maximum iteration number (*generation number*).

The encoding scheme plays a vital role to yield the effectiveness and the efficiency of the algorithm [7]. On the other hand, our DRC scheduling deals with three kinds of assignments: sequence of activity between jobs, operator to each activity, and machine to each job. We use a simple single-string as one chromosome containing *n* alleles that represent the sequence of jobs. In decoding phase, this sequence becomes a reference in resulting those three assignments to get the fitness value (the makespan). In the mathematical formulation, let $\bar{l} = \pi_{(i)}$ becomes the job on the *i*-th allele to represent a job sequence chromosome (see Fig, 2).

$\pi_{(1)}$	$\pi_{(2)}$	$\pi_{(3)}$	$\pi_{(4)}$	$\pi_{(5)}$
4	1	5	3	2

Fig. 2. An example of a chromosome represents a job sequence (4-1-5-3-2) with $n = 5$ jobs

Our decoding algorithm consists of seven steps (as follow) that need an iterative process as many as the number of activities, which is twice of the jobs number.

Procedure of decoding

1. Set parameters for the zero iteration ($y = 0$): setup number (z) = 0; assigned activity (b^*) = ∅; assigned job (l^*) = ∅; operator k expected moving completion time (TMW_k) = 0 $\forall \boldsymbol{k}$; assigned operator (k^*) = ∅; assigned machine (i^*) = ∅; unloading waiting list in machine i (WL_i) = ∅ $\forall \boldsymbol{i}$; busy machine matrix ($\boldsymbol{B}$) = ∅; operator k earliest assigning time (TW_k) = 0 $\forall \boldsymbol{k}$; operator k last machine location (L_k) = ∅ $\forall \boldsymbol{k}$ and machine i earliest assigning time (TM_i) = 0 $\forall \boldsymbol{i}$.
2. Add y by 1. If $y > 2n$ stop the iteration and compute makespan that is equal to the maximum value of TW_k of all k. Otherwise, continue to step 4.
3. If $\boldsymbol{z < n}$, set $i^* = i \; \exists \boldsymbol{i = 1, 2, \ldots, m \mid TM_i = \min_i TM_i}$. Otherwise, set $\boldsymbol{i^* = i \; \exists i \in B \mid TM_i = \min_{i \in B} TM_i}$
4. Compute expected moving completion time to machine i^* of all operator k using equation (17).
$$\boldsymbol{TMW_k = TW_k + m_{ij} \; \forall \; k = 1, 2, \ldots, w, \; i = L_k, \; j = i^*} \quad (17)$$
Then, set $k^* = k \; \forall \; \boldsymbol{k = 1, 2, \ldots, w \; if \; TMW_k = \min_k TMW_k}$ and update $\boldsymbol{L_{k^*} = i^*}$
5. Set $b^* = 1$ if $\boldsymbol{i^* \notin B \cup n_s < n}$. Otherwise, $b^* = 2$
6. If $b^* = 0$, add z by 1, $l^* = \boldsymbol{\pi_{(z)}}$, add j^* to the $\boldsymbol{B}$, and update $WL_{j^*} = l^*$. Otherwise, set $b^* = 2$, $l^* = WL_{j^*}$, remove j^* from the $\boldsymbol{B}$ and update $WL_{j^*} = ∅$.
7. Update TM_{i^*} and TW_{k^*} using equation (18) and (19) and back to step 2.
$$\boldsymbol{TM_{i^*} = \begin{cases} s_{l^*} + max\{TM_{i^*}, TMW_{k^*}\} + p_{l^*}, \; if \; b = 0 \\ u_{l^*} + max\{TM_{i^*}, TMW_{k^*}\}, \; otherwise \end{cases}} \quad (18)$$
$$\boldsymbol{TW_{k^*} = \begin{cases} s_{l^*} + max\{TM_{i^*}, TMW_{k^*}\}, \; if \; b = 0 \\ u_{l^*} + max\{TM_{i^*}, TMW_{k^*}\}, otherwise \end{cases}} \quad (19)$$

We use the binary tournament as our selection method since it is more efficient than rank method and more effective than roulette wheel method [11] [12]. Moreover, we choose two-point crossover that has been largely adopted in combinatorial problem and block swapping scheme [13] for our crossover and mutation schemes. Finally, the stop criterion is the number of generations.

5 Numerical examples and computational results

The objective of this benchmark is to compare the performance of the solution quality and run time between solver (SLV) using Gurobi 7.5.2, PGA, and random search (RS) which are executed on 16-GB RAM PC powered by an octa-core 3.6-GHz processor. The PGA and RS have been coded in RUBY® language. The RS uses same encoding and decoding scheme as in PGA to get the fitness value of a string. The stop criterion for RS is the number of evaluated strings as many as in PGA.

Our experiment consists three cases based on the number of jobs (*n*), machines (*m*), and operators (*w*) – small, medium, and big-sized problem (see Table 1). We generate all parameters from uniform distributions which are U[1, 79], U[1, 99], U[1, 20], and U[3, 10] respectively for setup, machining, unloading, and moving time. The solver is time limited in the medium-sized problem for 300 seconds and not run in the big-sized problem despite memory problem.

Table 1. Computational results

Case	Subcase	Gurobi	RPD Mean			Run Time Mean (sec.)		
	(*n* x *m* x *w*)	gap** (%)	SLV	PGA	RS	SLV	PGA	RS
Small-sized problem PGA parameter* [10; 20; 0.5; 0.1]	4 x 3 x 2	0.00	0.00	0.00	0.00	8.93	0.01	0.00
	4 x 4 x 2	0.00	0.00	0.75	0.75	1.49	0.01	0.01
	4 x 4 x 3	0.00	0.00	0.00	0.00	0.69	0.01	0.01
	5 x 3 x 2	0.00	0.00	0.00	0.00	116.99	0.01	0.01
	5 x 4 x 2	0.00	0.00	6.00	6.00	46.89	0.02	0.01
	5 x 4 x 3	0.00	0.00	0.00	0.00	637.52	0.01	0.01
	Average	**0.00**	**0.00**	**1.13**	**1.13**	**135.42**	**0.01**	**0.01**
Medium-sized problem PGA parameter* [50; 150; 0.5; 0.1]	9 x 4 x 2	55.36	20.43	0.65	1.29	300	0.60	0.58
	9 x 4 x 3	50.82	23.98	0.33	0.65	300	0.74	0.71
	9 x 5 x 2	52.53	13.26	0.93	1.00	300	0.61	0.58
	9 x 5 x 3	48.10	37.62	0.29	1.71	300	0.75	0.72
	10 x 4 x 2	66.87	56.21	1.03	2.07	300	0.66	0.63
	10 x 4 x 3	59.35	46.43	0.79	1.90	300	0.81	0.79
	10 x 5 x 2	70.36	74.48	0.76	1.52	300	0.67	0.65
	10 x 5 x 3	60.11	74.0	1.02	2.22	300	0.83	0.80
	Average	**57.94**	**43.31**	**0.72**	**1.55**	**300.00**	**0.71**	**0.68**
Big-Sized Problem PGA parameter* [150; 200; 0.8; 0.05]	100 x 15 x 7	-	-	0.47	5.02	-	75.47	68.69
	100 x 15 x 9	-	-	0.38	3.99	-	88.22	80.23
	100 x 20 x 7	-	-	0.80	3.57	-	77.75	70.69
	100 x 20 x 9	-	-	1.27	6.34	-	94.98	84.32
	Average	-	-	**0.73**	**4.73**	-	**84.10**	**75.98**

*PGA parameter respectively consists of generation number, population size, and crossover and mutation probability
** Gurobi gap measures the difference between its lower and upper bound

Before comparing, we set GA parameter using a full factorial experimental design without replication for each case and analyze the result using graphical descriptive statistics to find the convergence point that represents the optimal generation number and Tukey test with 95% confidence interval to find the least population size from the best. Then, both parameters are used in another full factorial experimental design to find the optimal crossover and mutation probability in each case.

The number of strings needed to reach convergence point becomes higher when the case becomes more complex. The algorithm needs only 200 strings in the small-sized problem and needs more strings of 7,500 and 30,000 strings respectively for medium and big-sized problems. On the other hand, the generation number is always smaller than the population size in all cases. Also, in our experiments, we acknowledge that the crossover probability never becomes lower when the case becomes more complex, which needs to be analyzed more deeply in the future.

We implement a full factorial experiment design with five replication for PGA and RS and without replication for the solver. To evaluate the solution quality, we use relative percentage deviation (RPD) for each method *i* according to the equation (20):

$$RPD_i = 100 \cdot \frac{TCT_i - TCT_i^{min}}{TCT_i^{min}} \tag{20}$$

Where TCT_i is the total completion time (makespan) resulted from method *i* and TCT_i^{min} is the best makespan among all methods.

Table 1 also shows that PGA and RS can yield the optimal solution in the small-sized problem except for subcase 4 x 4 x 2 and 5 x 4 x 2. We acknowledge that our decoding algorithm always assigns a resource immediately after it is free, but, in that subcase, the optimal solution has different sequence logic. Our meta-heuristics also have better computational time than the solver in the small-sized problem.

The PGA outperforms other methods for average RPD in the medium-sized problem respectively with only 0.72% compared to RS with 1.55% and solver with 43.31%. Also, in the big-sized problem, the average deviation of PGA solution from the best is only 0.73% compared to the RS with 4.73%. By running for 5 minutes, the Gurobi solver can only reach 57.94%, on average, of the difference between its upper and lower bounds in the medium-sized problem. We also acknowledge that the solver cannot reach the optimal solution after one day running. Moreover, the computational (run) time of PGA is very reasonable compared to the RS that the difference is not more than 15 seconds even in the big-sized problem.

6 Conclusions

In this paper, we propose a novel DRC scheduling under an identical parallel machine environment that considers operator working modes and moving activity concerning the makespan minimization objective. We develop a MILP model at first, but it can only solve the small-sized problem in reasonable time. Therefore, we propose a permutation-based genetic algorithm (PGA) to tackle the computational burden. To test its performance in solution quality and computational time, we compare the PGA to

the solver and RS. Firstly, we set the PGA parameter using statistical approaches. We acknowledge that the appropriate parameters depend on the case. The comparison results show that the PGA could solve the problem in a reasonable time that is faster than the solver with a good quality solution that is better than random search. There are several ideas for improvement for the next potential research. First, to involve a new social objective function such as operator's productivity or workload balance. Second, to extend the model into more complex machine configuration, e.g., flow shop or job shop. Third, to implement other meta-heuristics methods.

References

1. Baptiste, P., Rebaine, D., and Zouba, M.: FPTAS for the two identical parallel machine problem with a single operator under the free changing mode. European Journal of Operational Research 256, 55–61 (2017).
2. ElMaraghy, H., Patel, V., and Abdallah, I. B.: Scheduling of Manufacturing Systems under Dual-Resource Constraints Using Genetic Algorithms. Journal of Manufacturing Systems 19 (3), 186–201 (2000).
3. Hu, P.: Minimizing total flow time for the worker assignment scheduling problem in the identical parallel-machine models. International Journal of Advance Manufacturing Technology 25, 1046–1052 (2005).
4. Chaudry, I. A., and Drake, P. R.: Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machine using genetic algorithms. International Journal of Advance Manufacturing Technology 42, 581–594 (2009).
5. Agnetis, A., Flamini, M., Nicosia, G., and Pacifici, A.: A job-shop problem with one additional resource type. Journal of Scheduling 14, 225–237 (2011).
6. Paksi, A. B. N., and Ma'ruf, A.: Flexible Job-Shop Scheduling with Dual-Resource Constraints to Minimize Tardiness Using Genetic Algorithm. In: IOP Conference Series: Materials Science and Engineering 114 012060, Yogyakarta (2016).
7. Baptiste, P., Rebaine, D., and Zouba, M.: Solving the two identical parallel machine problem with a single operator. In: Framinan, J. M., Gonzalez, P. P., and Artiba, A. Proceedings of 2015 International Conference on Industrial Engineering and System Management (IESM), pp. 502–507. I^4e^2, Spain (2015).
8. Costa, A., Cappadonna, F. A., and Fichera, S.: A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times. International Journal of Advance Manufacturing Technology 69, 2799–2817 (2013).
9. Abedinnia, H., Glock, C. H., Groose, E. H., Schneider, M.: Machine scheduling problems in production: A tertiary study. Computers & Industrial Engineering 111, 403–416 (2017).
10. Holland, J. H.: Adaption in Natural and Artificial Systems. 1st ed. MIT Press, U.S. (1992).
11. Zhong, J., Hu, X., Gu, M., and Zhang, J.: Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms. In: International Conference on CIMCA-IAWTIC'05 (2005).
12. Razali, N. M., Geraghty, J.: Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. In: Proceedings of the World Congress on Engineering II, London (2011).
13. Costa, A., Cappadonna, F. A., and Fichera, S.: A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling problem. Journal of Intelligence Manufacturing 28, 1269–1283 (2017).